Exploration de la factorisation d'un modèle de classes sous contrôle des acteurs

André Miralles[*] — Xavier Dolques[**] — Marianne Huchard[***] — Florence Le Ber[**] — Thérèse Libourel [****] — Clémentine Nebut[***] — Abdoulkhader Osman-Guédi[*,***]**

[*] *Tetis/IRSTEA, Montpellier, France, prénom.nom@teledetection.fr*
[**] *ICube, Univ. de Strasbourg/ENGEES, CNRS, Strasbourg, France*
[***] *LIRMM, Univ. Montpellier 2 et CNRS, Montpellier, France*
[****] *Espace Dev, Montpellier, France*
[*****] *Université de Djibouti, Djibouti*

RÉSUMÉ. *Nous nous intéressons à la construction du modèle de classes d'un système d'information environnemental. Cette construction est réalisée dans le cadre d'un processus de conception multi-acteurs dont les intérêts divergent ou sont conflictuels et qui nécessite de travailler en groupes autour d'un modèle qu'on essaie d'améliorer progressivement. Cette amélioration progressive se fait notamment en étudiant les classes et les associations et en faisant émerger des classes et des associations d'un niveau d'abstraction plus élevé. La technique que nous utilisons est dérivée de l'Analyse Formelle de Concepts. La construction de toutes les abstractions en une seule étape produit des nouvelles classes et des nouvelles associations en nombre trop élevé pour l'analyse par les experts. Nous proposons dans cet article de réaliser une construction par étapes, sous le contrôle des acteurs, de ces nouvelles classes et associations. Cette solution est testée sur un modèle de système d'information environnemental.*

ABSTRACT. *We are involved in the building of the class model of an information system in the environment domain. This building is done in the framework of a many-actors design process. Actor concerns are various and sometimes conflicting, and progressively designing the model during working group sessions is required. This progressive improvement relies on the emergence of more abstract classes and associations. We use an optimization technique based on Formal Concept Analysis. Nevertheless, building all abstractions in a single step produces too much concepts to be analyzed by the domain experts. In this paper, we propose a step-by-step approach, under the actor control, to extract these new abstractions. This solution is experimented on an environmental information system.*

MOTS-CLÉS : *refactorisation, Analyse Formelle de Concepts, Pesticides*

KEYWORDS: *refactoring, Formal Concept Analysis, Pesticides*

1. Introduction

Dans les domaines d'intervention d'Irstea (environnement, territoire, biodiversité, etc.), le développement d'un système d'information suppose de capturer des connaissances de plus en plus complexes et en évolution impliquant différents types d'acteurs parmi lesquels des scientifiques. Dans ce contexte, l'analyse du système est une phase cruciale du développement affectant directement la qualité du système d'information car c'est au cours de cette phase que les concepts du domaine et les exigences des acteurs sont capturés.

En situation multi-acteurs, l'analyse du système est effectuée, soit avec l'ensemble des acteurs au cours de séances d'analyse, soit par groupes. La première méthode peut poser des problèmes car, lorsque certains acteurs ont des intérêts divergents, certains d'entre eux vont avoir tendance à orienter l'analyse à leur bénéfice ou à être minimaliste. Dans un cas comme dans l'autre, le résultat est que le modèle final sera affecté et ne répondra pas aux besoins. Sa qualité est altérée. Aussi, effectuer l'analyse avec des groupes d'acteurs homogènes va permettre d'éviter partiellement ces inconvénients. La difficulté pour le concepteur réside dans l'intégration des différents points de vue pour obtenir un modèle unique. Il est confronté à un problème d'intégration de schémas et notamment de duplication de concepts. Il n'est pas rare que des concepts majeurs soient dupliqués dans plusieurs schémas. Par exemple, dans le modèle SIE Pesticides (Miralles *et al.*, 2011 ; Vernier *et al.*, 2013), *Pesticide* est un concept majeur d'un sous-modèle *Activités Agricoles* car les agriculteurs utilisent les pesticides dans leur itinéraire technique. Il existera aussi dans un sous-modèle *Activités Métrologiques* car on retrouve des pesticides dans les échantillons d'eau prélevés dans les cours d'eau.

L'objectif des recherches en cours conduit à identifier automatiquement ces doublons (illustrés ici au niveau des classes mais qui apparaissent aussi au niveau des attributs, des rôles, des opérations ou des associations) afin de simplifier le modèle. Lors de l'identification des doublons, de nouvelles abstractions vont également émerger naturellement par factorisation de caractéristiques communes à des classes, des attributs, des rôles, des associations ou des opérations. Par exemple, la factorisation des attributs *TypeProduction* d'un éleveur et d'un viticulteur va faire émerger un nouveau concept *Agriculteur* généralisant *Éleveur* et *Viticulteur*.

Dans nos travaux, nous utilisons l'Analyse Relationnelle de Concepts (ARC), méthode d'analyse de données qui offre une solution exacte et unique pour la suppression des doublons. L'inconvénient de cette méthode est que le processus est combinatoire et peut générer un nombre de nouveaux concepts qu'il est humainement difficile, voire impossible, d'analyser (Guédi *et al.*, 2013).

Afin de maîtriser cette explosion combinatoire, nous proposons ici une approche de factorisation pas-à-pas qui introduit une partie seulement des nouvelles abstractions à chaque étape. Notre approche présente comme autre avantage de mettre sous contrôle des acteurs le processus de factorisation, ceux-ci ayant la possibilité de valider ou de rejeter les concepts émergeant au cours de l'étape. Nous parlerons de *chemin de*

factorisation pour désigner une succession d'étapes lors desquelles sont effectués des choix d'éléments et de relations pour avancer dans le processus. La Figure 1 illustre cette notion de *Chemin*.

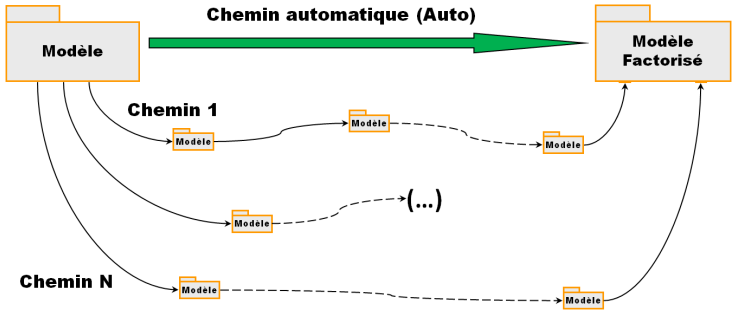

Figure 1. *Notion de chemins de factorisation*

La suite de l'article se présente de la manière suivante. Dans la section 2, nous présentons de manière succincte l'Analyse Relationnelle de Concepts appliquée aux modèles de classes et les problèmes identifiés dans diverses expériences menées précédemment. Dans la section 3, nous définissons la notion de chemin de factorisation. La section 4 est consacrée à la mise en œuvre de l'approche sur le modèle SIE Pesticides. Enfin nous présentons des travaux connexes dans la section 5, puis nous concluons dans la section 6.

2. Découverte d'abstraction par Analyse Relationnelle de Concepts

Une littérature assez riche montre comment la factorisation de modèles de classes articulés autour d'une hiérarchie de spécialisation/généralisation peut être mise en œuvre grâce à une méthode d'analyse de données basée sur la théorie des treillis (l'Analyse Formelle de Concepts (Ganter et Wille, 1999)). Une manière classique de l'appliquer consiste à décrire les classes par leurs caractéristiques (attributs et/ou opérations) dans une relation binaire (appelée contexte formel) (Godin et Mili, 1993). Les caractéristiques peuvent être décrites plus ou moins finement, et différentes formes de représentation dans le contexte formel peuvent être effectuées. De ce contexte formel peuvent être extraits des concepts, groupes maximaux de classes partageant des ensembles maximaux d'attributs et d'opérations. Ces concepts s'interprètent comme des classes, soit qu'ils représentent précisément une des classes de départ, soit qu'ils représentent une nouvelle super-classe introduite pour factoriser des caractéristiques communes. Les concepts sont organisés dans une structure de spécialisation appelée le treillis de concepts.

Dans la Figure 2(a), par exemple, l'attribut `name` se trouve dupliqué dans les classes `B1` et `B2` (Guédi *et al.*, 2013). Une méthode de factorisation basée sur l'AFC permettra d'obtenir le modèle de la Figure 2(b). Une nouvelle superclasse (désignée

par un expert `NamedElement`) est introduite automatiquement par l'AFC pour factoriser cet élément.

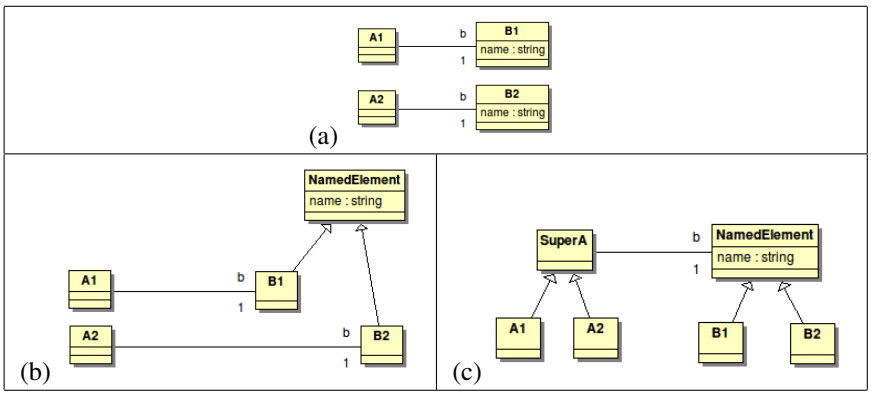

Figure 2. *Description du mécanisme de factorisation dans un modèle de classes*

Cependant les modèles de classes dans lesquels les attributs sont typés par des classes et dans lesquels on trouve également des associations ne peuvent être traités par ce modèle simple. Dans notre exemple, la méthode basée sur l'AFC ne calcule pas la factorisation plus complète présentée dans la Figure 2(c), dans laquelle une association généralise les deux associations de départ entre la nouvelle classe `SuperA` et la nouvelle classe `NamedElement`. Dans ce cas général, les données à traiter sont en effet constituées d'objets (classes, attributs, associations, rôles, opérations) décrits par des caractéristiques (leur nom, leur type s'il est primitif par exemple) mais également par des relations avec d'autres objets : une association possède des rôles, le rôle aboutit sur une classe, une classe possède un attribut, etc. L'Analyse Formelle de Concepts initiale a été étendue au cadre théorique de l'Analyse Relationnelle de Concepts (ARC, (Hacène *et al.*, 2013)) pour traiter de telles données et produire des factorisations plus systématiques et de portée plus grande comme celle de la Figure 2(c). Les données de l'ARC se présentent sous la forme d'une famille de contextes.

Définition 1. ***Famille relationnelle de contextes (RCF)***
Une famille relationnelle de contextes est un couple $(\mathbf{K}, \mathbf{R})$ *avec :*

– $\mathbf{K} = \{\mathcal{K}_i\}_{i=1,\ldots,n}$ *un ensemble de contextes formels* $\mathcal{K}_i = (G_i, M_i, I_i)$ *(relations objet-caractéristique), où* G_i *est l'ensemble des objets,* M_i *est l'ensemble des caractéristiques et* $I_i \subseteq G_i \times M_i$*.*

– $\mathbf{R} = \{r_j\}_{j=1,\ldots,m}$ *est un ensemble de relations objet-objet* r_j *où* $r_j \subseteq G_{i_1} \times G_{i_2}$ *pour* $i_1, i_2 \in \{1, \ldots, n\}$*.*

Dans le cadre de nos données, divers contextes formels et diverses relations objet-caractéristique peuvent être considérés. Par exemple une famille relationnelle de contextes peut se composer simplement de deux contextes formels respectivement pour les classes et les attributs ($\mathbf{K} = \{\mathcal{K}_{class}, \mathcal{K}_{attributs}\}$) et de deux relations ($\mathbf{R} =$

$\{r_{owns}, r_{hasType}\}$) indiquant quelle classe possède quel attribut et quel est le type d'un attribut. D'une famille relationnelle de contextes, l'ARC fait émerger itérativement des concepts. Lors de la première étape, un treillis de concepts est construit pour chaque contexte formel, par exemple un treillis de classes et un treillis d'attributs. Si les attributs sont décrits par leur nom dans le contexte formel $\mathcal{K}_{attribute}$, un concept de ce treillis groupe des attributs de même nom. Lors des étapes suivantes, les relations objet-objet sont intégrées sous la forme de caractéristiques relationnelles dans le contexte formel, en exploitant un opérateur de *scaling* (existentiel ou universel) et les concepts construits à l'étape précédente. Si une classe C est en relation avec un attribut a groupé dans un concept d'attributs C_a obtenu à l'étape précédente, on lui attribue la caractéristique relationnelle $\exists owns(C_a)$, pour indiquer que C possède l'un au moins des attributs groupés dans C_a.

Plusieurs expériences ont été menées par le passé, chacune s'appuyant sur une configuration (modèle de données) propre. Nous nous intéressons ici au point sensible qui a été relevé lors de ces expériences et qui est celui de la complexité pratique de l'approche. Pour simplifier, nous présentons seulement le nombre de concepts de classes car il s'agit de l'un des éléments de premier plan des modèles. Dans (Roume, 2004), on trouve une configuration assez complète dans laquelle sont pris en compte dans des contextes formels les classes, les attributs, les méthodes, les associations (décrites par des multiplicités), et les relations considérées sont association-classe, classe-attribut, attribut-classe et classe-méthode. Sur l'un des plus gros modèles étudiés (issu de France Télécom), formé de 57 classes, 167 concepts de classes apparaissent. Dans (Hacène, 2005), les contextes formels se limitent aux classes, propriétés (attributs et rôles) et les relations sont classe-propriété et propriété-classe. Avec cette description, sur un petit modèle de 6 classes (Jetsgo), 35 concepts de classes sont créés. Plus récemment, (Falleri *et al.*, 2008) ont utilisé une configuration identique et l'ont testée sur la librairie Apache Common Collections, où les 250 classes ont produit un nombre assez raisonnable de 284 concepts de classes. Sur le méta-modèle UML2, composé de 246 classes, l'approche produit le nombre fort élevé de 1780 concepts de classes. Ces quelques expériences, rapportées ici à grands traits, si elles montrent l'intérêt des concepts lors de l'analyse qualitative, font également ressortir les limites de l'approche. Les concepts de classes (ou des autres objets telles que les associations, les attributs, les opérations ou les rôles) issus de la méthode sont destinés à être examinés par des experts, et il semble difficile d'en présenter de tels nombres dans la plupart des cas. La section suivante s'attache à trouver une solution à ce problème.

3. Chemins de factorisation

Dans la partie précédente, nous avons montré comment l'ARC permettait de construire des formes normalisées de modèles de classes, mais également les limites qu'elle présente : une quantité potentiellement trop importante de nouvelles abstractions à présenter aux experts et des concepts extraits qui ne sont pas toujours utiles dans le domaine métier visé. Dans cette section, nous introduisons une approche exploratoire

appuyée sur l'ARC qui vise à réduire et à maîtriser ces limites. Nous proposons de contrôler l'ARC à différents niveaux, en nous arrêtant à chaque étape pour examiner quelles factorisations ont été réalisées et ainsi quelles abstractions sont apparues. A chaque étape, l'expert choisit les relations objet-objet et objet-caractéristique sur lesquelles il veut porter son attention, ainsi qu'un algorithme de construction associé à chaque table objet-caractéristique. La seule contrainte à ce choix est qu'une relation objet-objet ne peut être choisie que si une classification des objets de la cible de la relation a été calculée à une étape précédente (menant à la notion de chemin de factorisation valide que nous ne définirons pas ici pour des raisons de place). Deux sous-ordres des treillis vont être utilisés : les AOC-posets (Attribute-Object-Concept-posets) (Petersen, 2001), qui se définissent comme des sous-ordres des treillis restreints aux concepts introduisant un objet ou une caractéristique, et les icebergs(i), qui se limitent aux concepts couvrant un nombre d'objets supérieur à un seuil i choisi par un utilisateur.

Dans le contexte de la factorisation de modèles UML, l'opérateur de *scaling* est toujours existentiel. Pour en donner une intuition, prenons l'exemple de la relation qui associe à une classe ses attributs (`class_attribut`). Plusieurs attributs peuvent être groupés par exemple parce qu'ils portent le même nom. Quand on s'intéresse à la transformation de la relation `class_attribut` par l'opération de *scaling* et à son influence dans la création de groupes de classes, on ne s'intéresse ordinairement pas à trouver des groupes de classes qui ne vont partager que des attributs d'un groupe (par exemple que des attributs nommés "mesure", inférence qui serait réalisée par l'opérateur universel). On cherche plutôt à trouver des groupes de classes qui ont au moins un attribut du groupe (par exemple un attribut nommé "mesure", inférence qui serait réalisée par l'opérateur existentiel). Les choix réalisés successivement aux différentes étapes définissent ce que nous appellerons un chemin de factorisation. La notion de chemin de factorisation est inspirée de la notion de chemin d'exploration introduite pour l'utilisation de l'ARC dans le domaine de l'analyse de données (Dolques *et al.*, 2013).

Définition 2. ***Chemin de factorisation***
Pour une famille de contextes relationnels $(\mathbf{K}, \mathbf{R})$*, nous définissons une grammaire* (V_N, V_T, S, R) *pour décrire les chemins de factorisation.*
L'ensemble V_N *des non terminaux comprend les symboles entre '<' et '>'. L'ensemble* V_T *des terminaux est l'union des ensembles suivants : les entiers naturels* $\mathbb{N}$*, les relations objet-caractéristique* $\mathbf{K}$*, les relations objet-objet* $\mathbf{R}$*, les algorithmes* $\{AOC\text{-}poset, treillis, iceberg(i)\}$*, les opérateurs de scaling* $\{\exists, \forall\}$ *et les symboles* $\{\Rightarrow, [,], (,)\}$*. L'axiome* S *est* <$Chemin$>*. L'ensemble* R *des règles de production de* R *est le suivant :*

<$Chemin$> $\longrightarrow$ <$Etape$> | <$Etape$> <$Chemin$>
<$Etape$> $\longrightarrow$ <$NumeroEtape$> $\Rightarrow$ [<$DonneesEtape$>]
<$DonneesEtape$> $\longrightarrow$ <$PairesOC$> <$PairesOO$> |<$PairesOC$>
<$PairesOC$> $\longrightarrow$ <$PaireOC$> | <$PaireOC$> <$PairesOC$>
<$PaireOC$> $\longrightarrow$ <$RelationOC$> (<$Algorithme$>)
<$PairesOO$> $\longrightarrow$ <$PaireOO$> | <$PaireOO$> <$PairesOO$>

$<PaireOO> \longrightarrow <RelationOO> (<Operateur>)$

$<Algorithme> \longrightarrow AOC\text{-}poset \mid treillis \mid iceberg$

$<NumeroEtape> \longrightarrow i, i \in \mathbb{N}$ $\quad$ $<Operateur> \longrightarrow \exists \mid \forall$

$<RelationOC> \longrightarrow r, r \in \mathbf{K}$ $\quad$ $<RelationOO> \longrightarrow r, r \in \mathbf{R}$

Le chemin ci-dessous se compose d'une première étape où on ne considère que les classes et les attributs, classés dans un AOC-poset, et d'une seconde étape avec ces mêmes objets, classés dans un AOC-poset, ainsi que la relation qui associe à une classe ses attributs, avec l'opérateur de *scaling* existentiel :

$0 \Rightarrow [\mathcal{K}_{class}(AOC\text{-}poset)\ \mathcal{K}_{attribute}(AOC\text{-}poset)]$

$1 \Rightarrow [\mathcal{K}_{class}(AOC\text{-}poset)\ \mathcal{K}_{attribute}(AOC\text{-}poset)\ r_{class-attribute}(\exists)]$

4. Application sur le modèle SIE Pesticides

Dans cette section, nous décrivons la manière dont nous mettons en œuvre des chemins de factorisation pour le modèle SIE Pesticides. La section 4.1 décrit le méta-modèle utilisé, puis dans la section 4.2 nous définissons trois chemins de factorisation que nous comparons avec le chemin de factorisation classique (*Chemin Auto*) pour illustrer notre démarche. Ces quatre chemins sont ensuite analysés de manière qualitative sur un extrait du modèle dans la section 4.3. Dans la section 4.4, nous appliquons la même approche au modèle complet et nous étudions les résultats de manière quantitative afin de valider sa faisabilité pratique.

4.1. *Méta-modèle utilisé*

Le méta-modèle de données que nous avons choisi est présenté sur la Figure 3 (gauche). Les classes (contexte $\mathcal{K}_{class}$) y sont décrites par leur nom. On leur associe leurs attributs et leurs rôles (relations objet-objet $r_{class-attribute}$ et $r_{class-role}$). Les associations (contexte $\mathcal{K}_{association}$) sont décrites par leur nom. On leur associe leurs rôles (relation objet-objet $r_{association-role}$). Les attributs (contexte $\mathcal{K}_{attribute}$) sont considérés comme ayant des types primitifs (choix de modélisation dans le cadre de ce projet) et sont simplement décrits par leurs noms. Les rôles (contexte $\mathcal{K}_{role}$) sont décrits par leurs types qui sont des classes (relation objet-objet $r_{role-class}$). Les quelques opérations présentes (rares dans ce modèle destiné à l'analyse métier du système d'information) n'ont pas été considérées. Lorsque les rôles ne sont pas nommés, ils sont ignorés. Par ailleurs, lors de la conception du modèle de classes du SIE Pesticides, un soin particulier a été apporté à exprimer la navigation sur les associations. La Figure 3 (droite) présente un exemple de modèle composé de deux classes, munies d'attributs et de rôles dans une association (en haut) et sa traduction dans le méta-modèle choisi. Les noms des liens ne figurent pas car il n'y a pas d'ambiguïté sur leurs noms. On peut y noter que la classe $Person$ est connectée au rôle c, mais que la

classe Car n'est pas connectée au rôle p, pour traduire le sens de navigabilité. Cette représentation se rapproche de l'instanciation du méta-modèle UML classique.

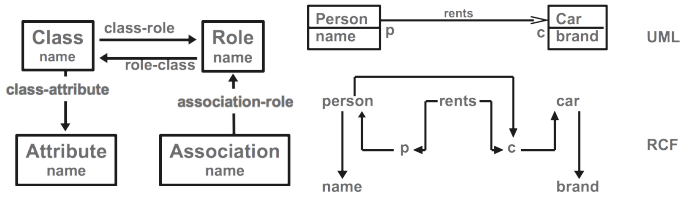

Figure 3. *Méta-modèle de données utilisé et modèle de classes (extrait) dans la représentation choisie*

4.2. *Chemins de factorisation*

Dans cette section, nous définissons quatre chemins de factorisation qui serviront à bâtir le cas d'étude. Le *Chemin Auto(matique)* (cf. Figure 1), utilisé jusqu'à présent par les utilisateurs de l'ARC, sera utilisé comme référence lors de l'analyse des résultats.

Chemin de factorisation. *Auto*

$0 \Rightarrow [\mathcal{K}_{class}(AOC\text{-}poset)\ \mathcal{K}_{attribute}(AOC\text{-}poset)\ \mathcal{K}_{role}(AOC\text{-}poset)\ \mathcal{K}_{association}(AOC\text{-}poset)]$

$i \Rightarrow [\mathcal{K}_{class}(AOC\text{-}poset)\ \mathcal{K}_{attribute}(AOC\text{-}poset)\ \mathcal{K}_{role}(AOC\text{-}poset)\ \mathcal{K}_{association}(AOC\text{-}poset)$

$r_{class-attribute}(\exists)\ r_{role-class}(\exists)\ r_{class-role}(\exists)\ r_{association-role}(\exists)], i \in [1, 6]$

Le premier chemin commence par classer les classes et les attributs, puis les classes d'après leurs attributs, puis les rôles d'après les classes, les classes d'après leurs rôles et enfin les associations d'après leurs rôles. Il faut noter que, dans ce chemin, il y a une rupture à l'étape 2 à laquelle les classes, précédemment décrites par leurs attributs, ne sont plus décrites par aucune information.

Chemin de factorisation. *1*

$0 \Rightarrow [\mathcal{K}_{class}(AOC\text{-}poset)\ \mathcal{K}_{attribute}(AOC\text{-}poset)]$

$1 \Rightarrow [\mathcal{K}_{class}(AOC\text{-}poset)\ \mathcal{K}_{attribute}(AOC\text{-}poset)\ r_{class-attribute}(\exists)]$

$2 \Rightarrow [\mathcal{K}_{role}(AOC\text{-}poset)\ \mathcal{K}_{class}(AOC\text{-}poset)\ r_{role-class}(\exists)]$

$3 \Rightarrow [\mathcal{K}_{class}(AOC\text{-}poset)\ \mathcal{K}_{role}(AOC\text{-}poset)\ r_{class-role}(\exists)]$

$4 \Rightarrow [\mathcal{K}_{association}(AOC\text{-}poset)\ \mathcal{K}_{role}(AOC\text{-}poset)\ r_{association-role}(\exists)]$

Dans les chemins 2 et 3, on cumule le long du chemin les abstractions apprises, la définition du chemin se faisant en ajoutant des relations ou des contextes à chaque étape. La différence entre les deux est que dans le Chemin 2 on commence la description des classes par les attributs tandis que dans le Chemin 3 on la commence par les rôles.

Chemin de factorisation. *2*

$0 \Rightarrow [\mathcal{K}_{class}(AOC\text{-}poset)\ \mathcal{K}_{attribute}(AOC\text{-}poset)]$

$1 \Rightarrow [\mathcal{K}_{class}(AOC\text{-}poset)\ \mathcal{K}_{attribute}(AOC\text{-}poset)\ r_{class-attribute}(\exists)]$
$2 \Rightarrow [\mathcal{K}_{class}(AOC\text{-}poset)\ \mathcal{K}_{attribute}(AOC\text{-}poset)\ \mathcal{K}_{role}(AOC\text{-}poset)$
$r_{class-attribute}(\exists)\ r_{role-class}(\exists)]$
$3 \Rightarrow [\mathcal{K}_{class}(AOC\text{-}poset)\ \mathcal{K}_{attribute}(AOC\text{-}poset)\ \mathcal{K}_{role}(AOC\text{-}poset)$
$r_{class-attribute}(\exists)\ r_{role-class}(\exists)\ r_{class-role}(\exists)]$
$4 \Rightarrow [\mathcal{K}_{class}(AOC\text{-}poset)\ \mathcal{K}_{attribute}(AOC\text{-}poset)\ \mathcal{K}_{role}(AOC\text{-}poset)\ \mathcal{K}_{association}(AOC\text{-}poset)$
$r_{class-attribute}(\exists)\ r_{role-class}(\exists)\ r_{class-role}(\exists)]$
$5 \Rightarrow [\mathcal{K}_{class}(AOC\text{-}poset)\ \mathcal{K}_{attribute}(AOC\text{-}poset)\ \mathcal{K}_{role}(AOC\text{-}poset)\ \mathcal{K}_{association}(AOC\text{-}poset)$
$r_{class-attribute}(\exists)\ r_{role-class}(\exists)\ r_{class-role}(\exists)\ r_{association-role}(\exists)]$

Chemin de factorisation. *3*

$0 \Rightarrow [\mathcal{K}_{class}(AOC\text{-}poset)\ \mathcal{K}_{role}(AOC\text{-}poset)]$
$1 \Rightarrow [\mathcal{K}_{class}(AOC\text{-}poset)\ \mathcal{K}_{role}(AOC\text{-}poset)\ r_{class-role}(\exists)]$
$2 \Rightarrow [\mathcal{K}_{class}(AOC\text{-}poset)\ \mathcal{K}_{role}(AOC\text{-}poset)\ \mathcal{K}_{attribute}(AOC\text{-}poset)\ r_{class-role}(\exists)]$
$3 \Rightarrow [\mathcal{K}_{class}(AOC\text{-}poset)\ \mathcal{K}_{role}(AOC\text{-}poset)\ \mathcal{K}_{attribute}(AOC\text{-}poset)$
$r_{class-role}(\exists)\ r_{class-attribute}(\exists)\ r_{role-class}(\exists)]$
$4 \Rightarrow [\mathcal{K}_{class}(AOC\text{-}poset)\ \mathcal{K}_{role}(AOC\text{-}poset)\ \mathcal{K}_{attribute}(AOC\text{-}poset)\ \mathcal{K}_{association}(AOC\text{-}poset)$
$r_{class-role}(\exists)\ r_{class-attribute}(\exists)\ r_{role-class}(\exists)]$
$5 \Rightarrow [\mathcal{K}_{class}(AOC\text{-}poset)\ \mathcal{K}_{role}(AOC\text{-}poset)\ \mathcal{K}_{attribute}(AOC\text{-}poset)\ \mathcal{K}_{association}(AOC\text{-}poset)$
$r_{class-role}(\exists)\ r_{class-attribute}(\exists)\ r_{role-class}(\exists)\ r_{association-role}(\exists)]$

4.3. *Étude qualitative sur le modèle Station Métrologique*

Afin d'évaluer et d'appréhender le comportement de l'ARC pour les quatre chemins décrits précédemment, une première expérimentation a été effectuée sur *Station Métrologique* (cf. Figure 4), sous-modèle décrivant les stations de mesures dans le modèle SIE Pesticides.

Analyse du Chemin Auto Ce chemin conduit à une factorisation maximale en six étapes. Le modèle obtenu correspond au modèle de la Figure 5. Le processus de factorisation de l'ARC a conduit à l'émergence des classes *MeasuringDevice*, *QualityInformation* et *Data* qui factorisent respectivement les attributs *DeviceType* et *DeviceNumber* pour la première, *CodeQuality* pour la seconde et *MeasuringDate* pour la troisième. En outre, l'ARC a permis la factorisation des associations *River Gauging*, *Groundwater Instrumentation* et *Rainfall Instrumentation* de la Figure 4 en une "super-association" *Instrumentation* (cf. Figure 5) entre les classes *MeasuringStation* et la super-classe *MeasuringDevice*. De même, *Water Level Information*, *Rainfall Information* et *Groundwater Information* sont abstraites par la "super-association" *Information* reliant la classe *MeasuringStation* et la super-classe *Data*. Du point de vue métier, ces deux super-associations ont un sens et remplacent dans le modèle final les associations dont elles sont issues. L'ARC produit également une super-association *Monitoring* entre les super-classes *MesuringDevice* et *Data* qui factorise les associations entre les classes (*Water Level Monitoring*, *Groundwater Monitoring* et *Rainfall*

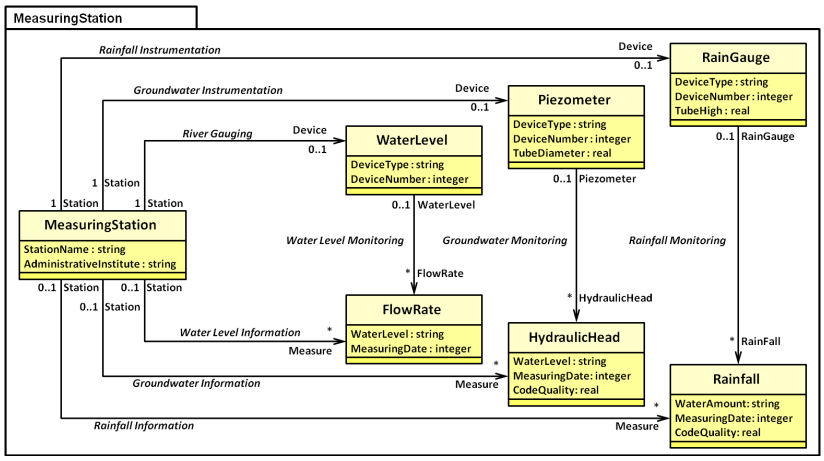

Figure 4. *Sous-modèle* Station Métrologique *(extrait du modèle SIE Pesticides)*

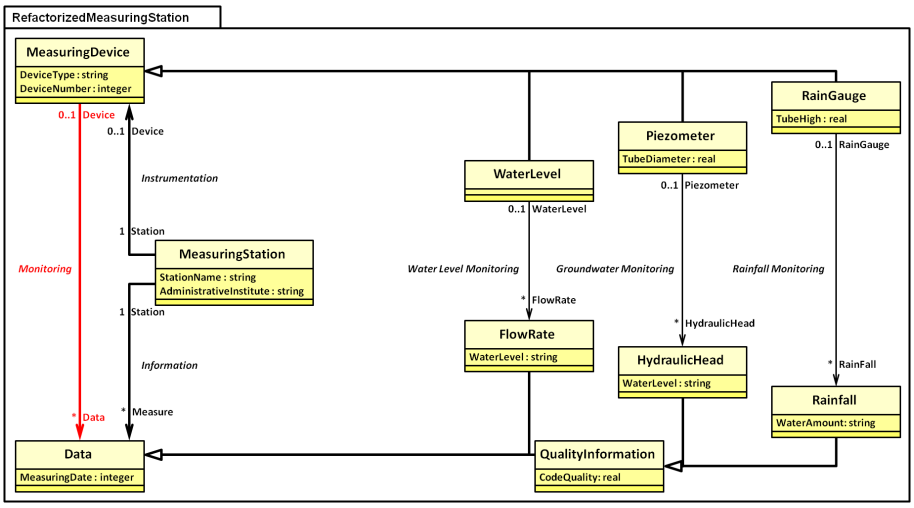

Figure 5. *Sous-modèle factorisé* Station Métrologique

Monitoring) décrivant les instruments de mesure et les données produites par ces instruments. Contrairement aux deux "super-associations" précédentes, cette dernière n'a pas un sens métier assez précis puisque, si elle était adoptée en remplacement des trois autres, elle permettrait à un informaticien peu familier du domaine d'associer le niveau du cours d'eau suivi à un pluviomètre. Par contre, elle est intéressante pour généraliser les associations de monitoring existantes et donner une vue plus générale du modèle. Les trois sous-associations doivent cependant rester pour la préciser (comme UML le permet).

Analyse du Chemin 1 À l'***étape 0***, les relations objet-objet ne peuvent pas être choisies car les treillis n'ont pas été construits. À ce stade, les classes sont à plat dans le treillis de classes et les attributs sont regroupés par leurs noms dans le treillis associé. À l'***étape 1***, les attributs de même nom sont groupés dans des concepts d'abstraction supérieure qui préfigurent en UML les super-classes *MeasuringDevice* et *Data*. À l'***étape 2***, les classes repartent à plat puisque le treillis de classes de l'étape précédente n'est pas pris en compte pour la construction de ce treillis. Les rôles sont quant à eux groupés d'après leurs noms mais aussi d'après le treillis des classes de l'étape d'avant. De ce fait, les rôles *Device* sont factorisés dans une abstraction qui préfigure la super-classe *MeasuringDevice* identifiée à l'étape précédente. De façon analogue, les rôles *Measure* sont factorisés dans une abstraction représentant *Data*. À l'***étape 3***, le treillis de classes à plat de l'étape précédente est ici hiérarchiquement réorganisé par le treillis des rôles précédent. Par exemple, les classes *WaterLevel*, *Piezometer* et *RainGauge* sont ainsi factorisées dans une abstraction prélude du *MeasuringDevice*. À l'***étape 4***, les associations navigables vers des instruments de mesure sont factorisées par le fait que les rôles sont nommés *Device*. Il en est de même pour les associations vers les données puisque le nom de leurs rôles est *Measure*. Contrairement au Chemin Auto, la courbe d'évolution du processus de "factorisation" du Chemin 1 n'est pas asymptotique. Cela s'explique par le fait que, pour une étape donnée, les treillis de l'étape précédente ne sont pas toujours pris en compte dans le calcul. Aussi, la factorisation maximale ne sera donc jamais atteinte.

Analyse du Chemin 2 À l'***étape 0***, comme la relation classe-attribut n'est pas prise en compte dans le calcul, le treillis des classes est plat et celui des attributs aussi. Les attributs de même nom *DeviceNumber*, *DeviceType*, *CodeQuality* et *MeasuringDate* sont factorisés au sein d'abstractions. À l'***étape 1***, le treillis des attributs factorisant les attributs de même nom affecte le treillis des classes où sont matérialisées les super-classes (*MeasuringDevice*, *QualityInformation* et *Data*) regroupant ces attributs. À l'***étape 2***, l'enrichissement du chemin avec la caractéristique rôle n'a aucune influence sur les treillis des classes et des attributs. Celui des rôles laisse entrevoir les futures factorisations issues des rôles de même nom (*Device* et *Measure* en particulier). À l'***étape 3***, l'introduction des relations classe-rôle et rôle-classe n'a pas d'effet majeur en matière de factorisation des classes. À l'***étape 4***, l'ajout au chemin de la caractéristique association n'a aucune influence en matière de factorisation supplémentaire sur les treillis de classes, d'attributs et de rôles. On constate que le treillis des associations est plat. À l'***étape 4***, la nouvelle relation association-rôle du chemin provoque la factorisation des associations au sein du treillis des rôles mais n'a aucun effet sur les trois autres treillis.

Analyse du Chemin 3 À l'***étape 0***, comme aucune relation classe-rôle n'est définie, le treillis des classes est plat, tout comme celui des rôles. Toutefois, les rôles *Device* sont réunis au sein d'un même contexte relationnel tout comme les rôles *Measure*. À l'***étape 1***, le treillis des classes est juste enrichi par les concepts de l'étape 0 factorisant les rôles. À l'***étape 2***, l'introduction des attributs comme caractéristiques ne provoque aucun changement dans les treillis des classes et des rôles. Dans le treillis des attributs, il est possible de constater les regroupements d'une part de *DeviceNum-*

ber et de *DeviceType* qui préfigurent l'abstraction *MeasuringDevice* et, d'autre part, de *CodeQuality* qui donnera la classe *QualityInformation*. La factorisation de l'attribut *MeasuringDate* générera la super-classe *Data*. À l'***étape 3***, les abstractions qui donneront en UML les classes *MeasuringDevice*, *QualityInformation* et *Data* sont matérialisées dans le treillis des classes. À l'***étape 4***, les associations sont introduites mais aucune nouvelle factorisation n'est observée. Le treillis des associations est plat. À l'***étape 5***, les abstractions qui représenteront les associations *Instrumentation*, *Information* et *Monitoring* apparaissent, chacune d'elles factorisant trois associations. Au-delà de cette étape, le processus est effectué automatiquement et les treillis ne sont plus analysés. Comme le processus prend en compte tous les treillis de l'étape précédente, son évolution est semblable au Chemin Auto mais avec un certain retard.

4.4. *Étude quantitative sur le modèle SIE Pesticides*

Dans cette section, nous appliquons l'approche sur le modèle complet et nous mesurons la quantité d'information à traiter par les experts à chaque étape. Nous souhaitons évaluer si la méthode est devenue faisable en pratique comparativement aux approches actuelles en une étape. La version 14 du modèle SIE Pesticides que nous étudions ici est composée de 171 classes ayant au total 130 attributs et 83 associations dont 78 rôles dans le sens de la navigabilité sont nommés et 5 n'ont pas de noms. C'est un modèle dont la connaissance métier du concepteur et la maîtrise du langage UML par les scientifiques impliqués dans le projet ont permis d'organiser sous forme de hiérarchie de nombreuses classes, limitant ainsi le nombre d'attributs par classes mais aussi le nombre d'associations. Ces modèles d'analyse sont créés avec la terminologie des scientifiques et les modèles d'implémentation dérivent de ces derniers par application systématique de transformations dont l'une est renommage des concepts.

Outre le chemin d'exploration automatique, nous avons appliqué à ce modèle les trois chemins décrits ci-dessus. Jusqu'aux étapes 4, pour le Chemin 1, et 5, pour les chemins 2 et 3, les caractéristiques d'entrée de l'algorithme sont celles décrites ci-dessus. Pour les chemins de factorisation 2 et 3, le processus est ensuite effectué automatiquement. La Figure 6 représente, pour chacun des quatre chemins, les évolutions du nombre total des concepts existants et nouveaux à chaque étape.

Nous observons que la courbe du chemin automatique est systématiquement située au-dessus des autres, surtout au cours des premières étapes. Comme tout processus de calcul itératif, son évolution est asymptotique vers un maximum qui sera atteint lorsque la factorisation maximale sera atteinte. Pour le Chemin 1, après une augmentation du nombre total de concepts à l'étape 1, cet indicateur décroît fortement. Cela est dû au fait que, à chaque étape, les caractéristiques d'entrée de l'ARC sont redéfinies sans prendre en compte le niveau de factorisation de l'étape précédente. Cette courbe montre que le processus n'est pas asymptotique et, de ce fait, il est impossible d'obtenir la factorisation maximale des concepts. C'est la raison pour laquelle le processus automatique n'a pas été lancé. Ce chemin a été abandonné pour le reste de l'analyse. Suite à ce constat, les chemins 2 et 3 ont été définis pour que le processus itératif soit

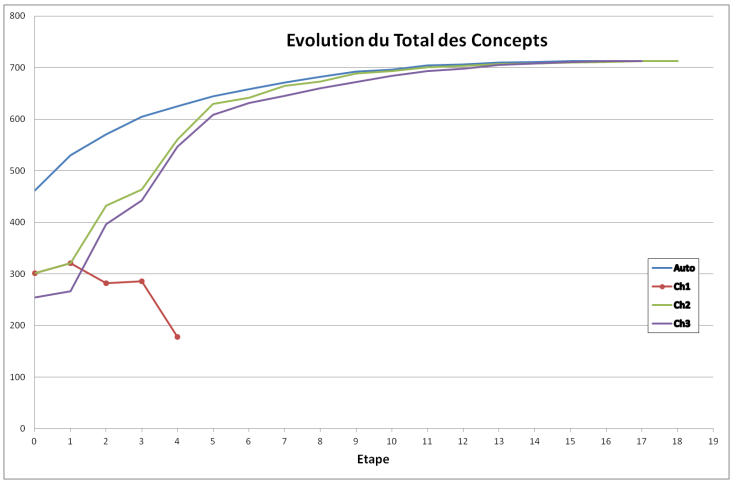

Figure 6. *Évolution du nombre total des Concepts à chaque étape*

cumulatif. Les caractéristiques et les treillis de l'étape n-1 sont pris en compte pour la factorisation de l'étape n. De ce fait, tout comme le chemin automatique, leurs courbes ont une évolution asymptotique et le nombre total de concepts (existants+nouveaux) final est le même pour ces trois chemins (cf. Tableau 1).

Step	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Auto	462	530	571	605	625	644	658	671	682	692	696	704	706	710	711	713	713		
Ch 2	301	321	432	464	560	630	642	665	673	689	693	701	703	707	708	710	711	713	713
Ch 3	254	266	396	442	547	608	631	645	660	672	684	693	698	705	708	711	713	713	

Tableau 1. *Évolution du nombre total de Concepts à chaque étape*

Les chemins 2 et 3 impliquant un nombre de concepts inférieurs à l'étape 0, ils ont un potentiel de factorisation moindre au cours des premières étapes. Toutefois, ces courbes montrent un saut assez marqué entre les étapes 1 et 2. La courbe du Chemin 3 est systématiquement au-dessous des autres. Pour autant, il nécessite une étape de moins que le Chemin 2 (cf. Tableau 1). Étant donné que l'objectif recherché est de limiter le nombre de nouveaux concepts à chaque étape, les Tableau 2 et Tableau 3 montrent cette évolution pour le nombre total des concepts et nombre des classes.

Transition	0->1	1->2	2->3	3->4	4->5	5->6	6->7	7->8	8->9	9->10	10->11	11->12	12->13	13->14	14->15	15->16	16->17	17->18
Auto	68	41	34	20	19	14	13	11	10	4	8	2	4	1	2	0		
Ch 2	20	111	32	96	70	12	23	8	16	4	8	2	4	1	2	1	2	0
Ch 3	12	130	46	105	61	23	14	15	12	12	9	5	7	3	3	2	0	

Tableau 2. *Évolution du nombre total de Concepts*

Le plus grand nombre total de concepts produits (130) est obtenu entre les étapes 1 et 2 du Chemin 3 (cf. Tableau 2), quasiment le double du maximum du chemin

Transition	0->1	1->2	2->3	3->4	4->5	5->6	6->7	7->8	8->9	9->10	10->11	11->12	12->13	13->14	14->15	15->16	16->17	17->18
Auto	32	13	12	6	7	4	5	3	5	0	4	0	2	0	1	0		
Ch 2	20	0	32	0	15	0	11	0	8	0	4	0	2	0	1	0	1	0
Ch 3	12	0	20	18	7	9	4	5	4	4	4	1	3	1	1	1	0	

Tableau 3. *Évolution du nombre de Classes*

automatique (68). Ceci s'explique par le fait que, le nombre maximal de concepts à atteindre (713) est constant et ce quel que soit le chemin. En effet, les chemins 2 et 3 démarrant avec un nombre de concepts inférieurs, le processus "rattrape" son retard de façon plus "brutale". Toutefois, le nombre de nouvelles classes entre deux étapes est mieux réparti et inférieur pour le Chemin 3 que pour les autres chemins (cf. Tableau 3). Pour les chemins Auto et 2, le plus grand nombre de nouvelles classes est 32 alors que, pour le Chemin 3, il est 20. L'inconvénient est, qu'après l'étape 5, il reste encore 37 classes à découvrir pour le Chemin 3 et seulement 24 et 27 respectivement pour les chemins Auto et 2. Une analyse métier approfondie des classes restant à découvrir après l'étape 5 pour les chemins 2 et 3 doit être menée avec les scientifiques impliqués dans le projet SIE Pesticides afin d'évaluer leur intérêt métier.

5. Travaux connexes

L'élimination des doublons est un sujet qui revêt différentes formes dans le domaine de la modélisation comme dans celui de la programmation. C'est notamment l'une des opérations de *refactoring* les plus connues (Fowler, 1999 ; Opdyke et Jonhson, 1993) et un cas particulier de la notion de clones dans le code (Roy *et al.*, 2009). Elle est au cœur d'approches visant à la reconstruction de hiérarchies de classes, de manière globale (Casais, 1991 ; Cook, 1992 ; Cherfi et Lammari, 2002) comme de manière incrémentale (Bergstein et Lieberherr, 1991). Elle s'accompagne de l'extraction d'abstractions organisées dans une hiérarchie de spécialisation, comme le développent certaines approches de rétro-ingénierie de modèles de bases de données relationnelles vers des modèles conceptuels (Akoka *et al.*, 1999).

L'Analyse Formelle de Concepts et les treillis en général ont été largement utilisés dans ce même contexte. À notre connaissance, les premiers usages des treillis datent de travaux dans le domaine de la refactorisation des schémas de bases de données orientés objets (Missikoff et Scholl, 1989 ; Rundensteiner, 1992). Dans le domaine de la programmation, l'AFC a été mise en œuvre pour extraire des interfaces abstraites à partir d'une hiérarchie de classes Smalltalk (Godin et Mili, 1993) ou pour refactoriser une hiérarchie de classes (Dicky *et al.*, 1996). Nous avons présenté dans la section 2 plusieurs approches utilisant l'ARC dans ce même domaine. Cet article poursuit dans cette voie, mais en tâchant d'améliorer sa faisabilité pratique et en travaillant par étapes plutôt que par analyse d'un résultat global.

6. Conclusion

Dans cet article, nous proposons une nouvelle approche afin de contrôler le processus de factorisation de certaines caractéristiques (classe, attribut, rôle et association) au sein d'un modèle de classes. Pour des modèles de taille conséquente, l'utilisation classique de l'Analyse Relationnelle de Concepts produit souvent un nombre important de nouveaux concepts qui rend difficile leur analyse métier par un expert du domaine. Nous simplifions cette analyse en limitant le nombre de concepts apparaissant à chaque étape du processus itératif par un choix de chemin de factorisation. Les résultats de factorisation sont comparés au processus classique. Les premiers résultats montrent que les chemins ne peuvent pas être choisis au hasard et que, si on souhaite atteindre une factorisation maximale sans diverger, il faut ajouter les caractéristiques de façon cumulative. L'objectif de réduire le nombre de concepts par étape est atteint dans les premières étapes contrôlées manuellement mais, pour atteindre la factorisation maximale, le processus va soit procéder par un saut à une étape ultérieure soit répartir le différentiel tout le long du chemin. Dans ce second cas, il est plus facile à un expert d'analyser les nouveaux concepts issus de la factorisation. Au stade actuel, il est impossible de prévoir l'un ou l'autre de ces deux comportements. Une expérimentation plus systématique sera menée pour cela.

Finalement, il faut noter que l'ordre d'introduction des caractéristiques modifie l'ordre des factorisations et que la reconstruction du modèle UML à chaque étape pourrait accélérer la convergence vers la factorisation maximale d'autant plus que certaines factorisations ne sont pas intéressantes du point de vue métier. La méthode d'analyse ARC est très sensible à l'orthographe, à la sémantique (polysémie, synonymie, parasynonymie...), etc. Cela constitue une limite de l'approche actuelle. Aussi, nous envisageons dans une prochaine étape d'enrichir les données d'entrée de l'ARC et de "piloter" ces méthodes par des ontologies métiers ou des techniques de Traitement Automatique du Langage Naturel. L'introduction des ontologies facilitera en particulier le nommage des nouveaux concepts générés et évitera en partie une expertise humaine.

Remerciements

Ce travail a été financé par l'ANR11_MONU14 Fresqueau et le projet Miriphyque du programme Pesticides du MEEDDM.

7. Bibliographie

Akoka J., Comyn-Wattiau I., Lammari N., « Relational Database Reverse Engineering : Elicitation of Generalization Hierarchies », *ER (Workshops)*, 1999, p. 173-185.

Bergstein P., Lieberherr K., « Incremental Class Dictionnary Learning and Optimization », *ECOOP'91*, 1991, p. 371–396.

Casais E., « Managing Evolution in Object Oriented Environments : An Algorithmic Approach », Thèse de doctorat, Université de Genève, 1991.

Cherfi S. S.-S., Lammari N., « Towards and Assisted Reorganization of Is-A Hierarchies », *Object-Oriented Inf. Systems*, vol. 2425 de *LNCS*, Springer-Verlag, 2002, p. 536–548.

Cook W., « Interfaces and Specifications for the Smalltalk-80 Collection Classes », *OOPSLA'92*, 1992, p. 1–15.

Dicky H., Dony C., Huchard M., Libourel T., « On Automatic Class Insertion with Overloading », *OOPSLA 96*, 1996, p. 251–267.

Dolques X., Ber F. L., Huchard M., Nebut C., « Analyse Relationnelle de Concepts pour l'exploration de données relationnelles », *EGC*, 2013, p. 121-132.

Falleri J.-R., Huchard M., Nebut C., « A Generic Approach for Class Model Normalization », *ASE 2008*, 2008, p. 431-434.

Fowler M., *Refactoring : Improving the Design of Existing Code*, Add.-Wesley Prof., 1999.

Ganter B., Wille R., *Formal Concept Analysis : Mathematical Foundation*, Springer-Verlag Berlin, 1999.

Godin R., Mili H., « Building and Maintaining Analysis-Level Class Hierarchies Using Galois Lattices », *OOPSLA 93*, 1993, p. 394–410.

Guédi A. O., Miralles A., Huchard M., Nebut C., « A Practical Application of Relational Concept Analysis to Class Model Factorization : Lessons Learned from a Thematic Information System », *Concept Lattices and Their Applications (CLA 2013)*, 2013, p. 9-20.

Hacène M. R., Huchard M., Napoli A., Valtchev P., « Relational concept analysis : mining concept lattices from multi-relational data », *Ann. Math. Artif. Intell.*, vol. 67, n° 1, 2013, p. 81-108.

Hacène M. R., « Relational concept analysis, application to software re-engineering », Thèse de Doctorat, Université du Québec À Montreal, 2005.

Miralles A., Pinet F., Carluer N., Vernier F., Bimonte S., Lauvernet C., Gouy V., « EIS-Pesticide : an information system for data and knowledge capitalization and analysis », *Euraqua-PEER Scientific Conference, 26/10/2011 - 28/10/2011*, Montpellier, FRA, 2011.

Missikoff M., Scholl M., « An Algorithm for Insertion into a Lattice : Application to Type Classification », *Proceedings of the 3rd Int. Conf. FODD'89*, , 1989, p. 64–82.

Opdyke W., Jonhson R., « Creating Abstract Superclasses by Refactoring », *Proc. of the 21st Annual Conference on Computer Science, Indianapolis (IN), USA*, ACM Press, New York (NY), USA, 1993, p. 66–72.

Petersen W., « A Set-Theoretical Approach for the Induction of Inheritance Hierarchies », *Proc. of FG/MOL-01), ENTCS*, vol. 53, Elsevier, July 2001, p. 296-308.

Roume C., « Analyse et restructuration de hiérarchies de classes », Thèse de Doctorat, Université Montpellier 2, 2004.

Roy C. K., Cordy J. R., Koschke R., « Comparison and evaluation of code clone detection techniques and tools : A qualitative approach », *Sci. Comp.. Prog.*, vol. 74, n° 7, 2009, p. 470-495.

Rundensteiner E. A., « A Class Classification Algorithm For Supporting Consistent Object Views », rapport, 1992, University of Michigan.

Vernier F., Miralles A., Pinet F., Carluer N., Gouy V., Molla G., vin Petit K., « EIS Pesticides : An environmental information system to characterize agricultural activities and calculate agro-environmental indicators at embedded watershed scales », *Agricultural Systems*, vol. 122, 2013, p. 11-21.